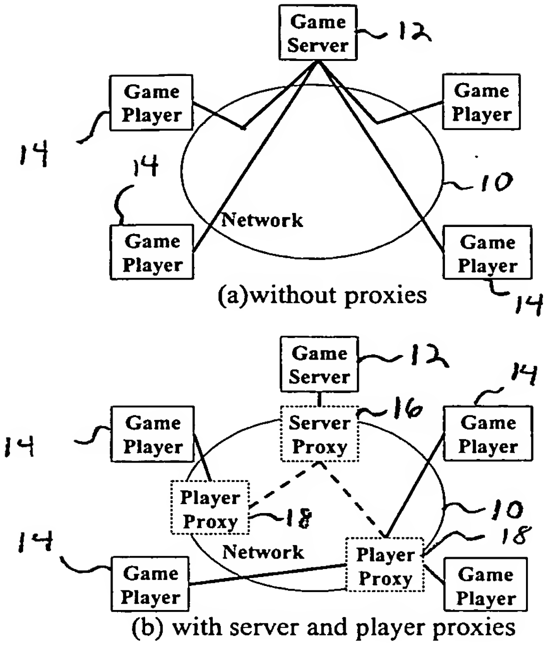

Fig. 1. Distributed game environment.

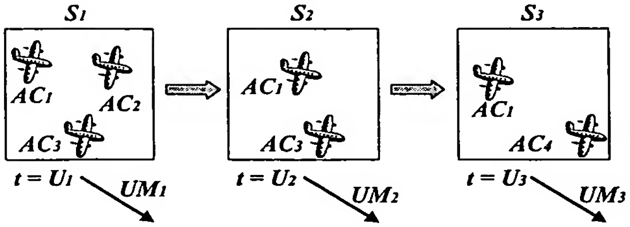

Fig. 2. Example of a state and its transitions.

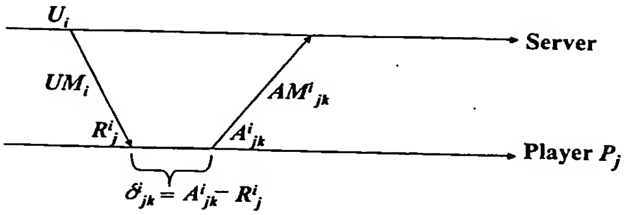

Fig. 3. Message exchange between server and players.

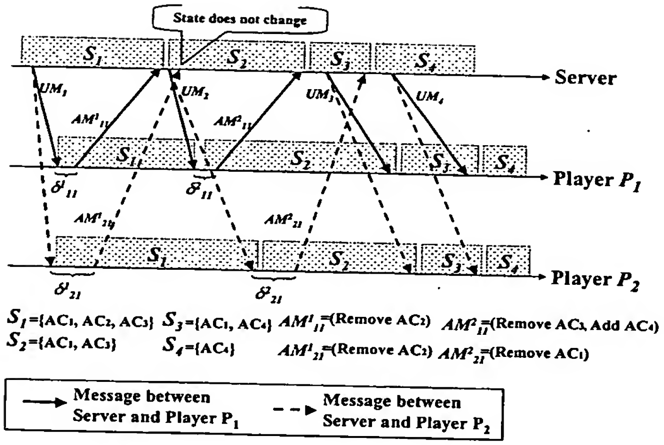
State does not change
S_1
S_2
S_3
S_4
Server
UM_1
AM^1_{11}
UM_2
AM^2_{11}
UM_3
UM_4
S_1
S_2
S_3
S_4
Player P_1
δ_{11}
δ_{11}
AM^1_{21}
AM^2_{21}
S_1
S_2
S_3
S_4
Player P_2
δ_{21}
δ_{21}
S_1={AC1, AC2, AC3} S_3={AC1, AC4} AM^1_{11}=(Remove AC2) AM^2_{11}=(Remove AC3, Add AC4)
S_2={AC1, AC3} S_4={AC4} AM^1_{21}=(Remove AC2) AM^2_{21}=(Remove AC1)
Message between Server and Player P_1
Message between Server and Player P_2

Fig. 4. Fair-order message delivery for state transitions shown in Figure 2.

```
Algorithm Fair-order Message Queueing(action message $M_k$):
Compute $D(M_k)$ = Delivery time of $M_k$;
Insert $M_k$ into Delivery Queue sorted according to $D(M_k)$;
If (Delivery Queue Size $> 1$)
    Recompute delivery time of existing messages;
```

Fig. 5. Algorithm for fair-order message queueing.

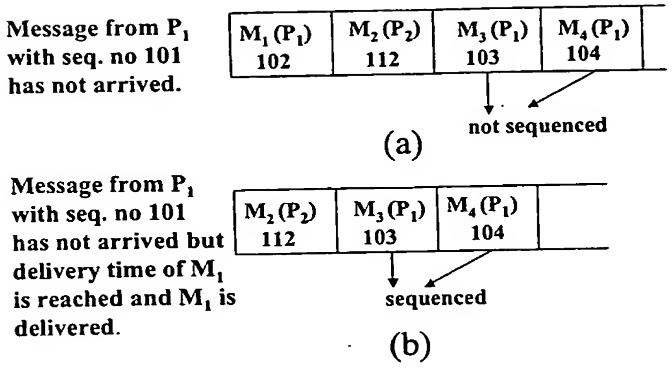
Message from P_1 with seq. no 101 has not arrived.
$M_1(P_1)$ 102
$M_2(P_2)$ 112
$M_3(P_1)$ 103
$M_4(P_1)$ 104
not sequenced
(a)
Message from P_1 with seq. no 101 has not arrived but delivery time of M_1 is reached and M_1 is delivered.
$M_2(P_2)$ 112
$M_3(P_1)$ 103
$M_4(P_1)$ 104
sequenced
(b)

Fig. 6. Example where messages arrive after their wait timeout.

```
Algorithm Fair-order Message Dequeuing(action message $M_k$):
  Delivery $M_k$ at $D(M_k)$;
  If (Delivery Queue Size > 1)
      Recompute delivery time of existing messages;
```

Fig. 7. Additional algorithm for fair-order message dequeuing when messages do not arrive within their wait timeout.

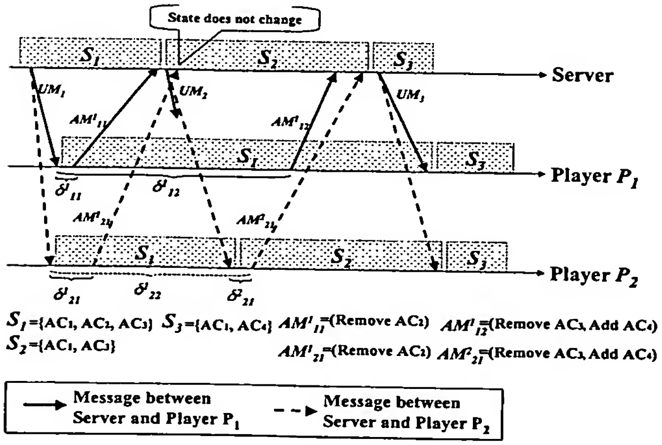
State does not change
S_1
S_2
S_3
Server
UM_1
UM_2
UM_3
AM^1_{11}
AM^1_{12}
S_1
S_3
Player P_1
δ^1_{11}
δ^1_{12}
AM^1_{21}
AM^2_{21}
S_1
S_2
S_3
Player P_2
δ^1_{21}
δ^1_{22}
δ^2_{21}
S_1={AC1, AC2, AC3} S_3={AC1, AC4} AM^1_{11}=(Remove AC2) AM^1_{12}=(Remove AC3, Add AC4)
S_2={AC1, AC3} AM^1_{21}=(Remove AC2) AM^2_{21}=(Remove AC3, Add AC4)
Message between Server and Player P_1
Message between Server and Player P_2

Fig. 8. Example of an inconsistent view of the game between two players, where the sequence of state changes at the server is shown in Figure 2.

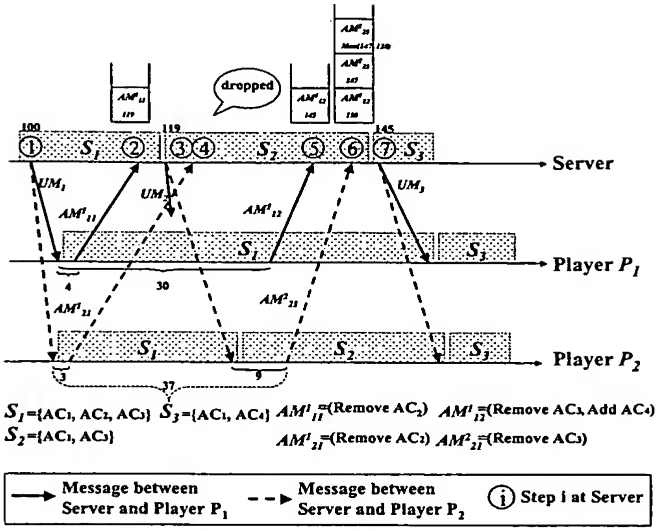
dropped
100
119
145
① S_1 ②
③④ S_2 ⑤ ⑥
⑦ S_3
Server
UM_1
UM_2
UM_3
AM^1_{11}
AM^1_{12}
AM^1_{21}
AM^2_{21}
S_1
S_3
Player P_1
4
30
S_1
S_2
S_3
Player P_2
3
37
9
S_1={AC1, AC2, AC3} S_3={AC1, AC4} AM^1_{11}=(Remove AC2) AM^1_{12}=(Remove AC3, Add AC4)
S_2={AC1, AC3} AM^1_{21}=(Remove AC2) AM^2_{21}=(Remove AC3)
Message between Server and Player P_1
Message between Server and Player P_2
ⓘ Step i at Server

Fig. 9. Example showing the fair-order message delivery algorithm.

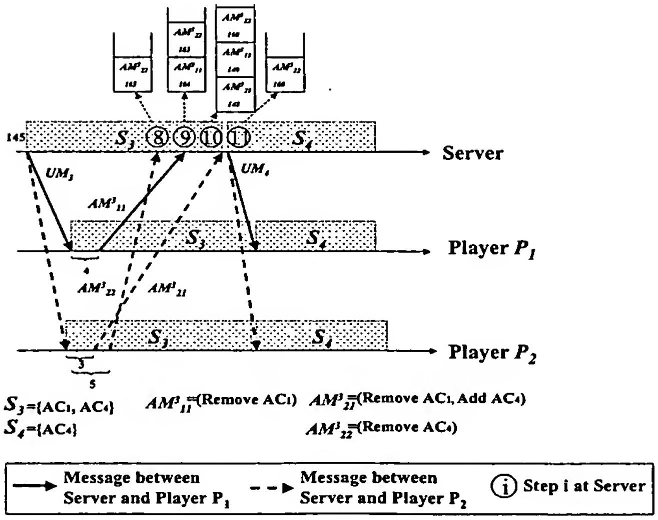
145
S_3 ⑧⑨⑩⑪ S_4
Server
UM_3
UM_4
AM^3_{11}
S_3
S_4
Player P_1
4
AM^3_{22}
AM^3_{21}
S_3
S_4
Player P_2
3
5
S_3={AC1, AC4} AM^3_{11}=(Remove AC1) AM^3_{21}=(Remove AC1, Add AC4)
S_4={AC4} AM^3_{22}=(Remove AC4)
Message between Server and Player P_1
Message between Server and Player P_2
ⓘ Step i at Server

Fig. 10. Example showing the fair-order message delivery algorithm with out-of-order message reception.

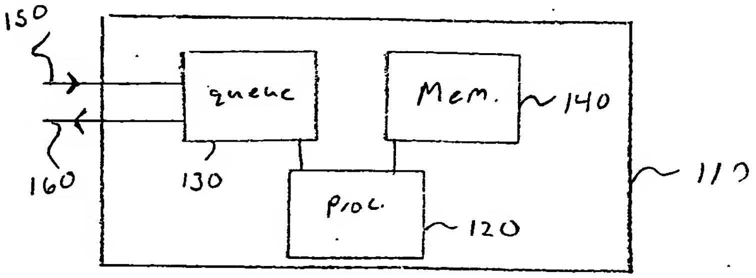

Fig. 11